Virtual Machine Logbook: Diploma Project Description

Andrea Cavalli, Julien Poffet (HEFR),
Paolo Calafiura and Yushu Yao (LBNL)

July 29, 2008

Abstract

We propose the development of a Virtual Machine Logbook (VML) application, designed to meet the requirements of the physicists and software developers of the ATLAS experiment at CERN[1]

1 ATLAS Software

The offline and high-level trigger software for the ATLAS experiment use the Athena/Gaudi component architecture [2]: using a python UI the physicists can assemble a job choosing from hundreds of C++ components. Some of these components called *Algorithms* implement a particular strategy to process Event data, say to find all Jets in the calorimeter. Other components called *Services* and *AlgTools* assist the Algorithms performing their tasks, providing for example access to calorimeter geometry, or organizing and filtering logging messages.

ATLAS Software consists of about 5M lines of code[1], organized in about 1000 packages[2]. Each package has a (cvs) tag number that distinguish different versions. A set of tags validated for a certain milestone (e.g. the simulation of events at 10TeV collision energy, or the online processing of events in 2008) defines a software release. Besides validated releases, developers use nightly releases built every day snapshotting the "collected" tag of each package in each project (usually the highest number tag). Each nightly build is kept for a week.

ATLAS software is currently supported only one one platform[3]: Scientific Linux CERN 4 using gcc 3.4.3. ATLAS software is distributed and installed semi-automatically using Pacman kits[3].

[1] mainly C++ and python

[2] the packages are further organized in a hierarchy of projects, but this is not important for this discussion.

[3] although builds for other platforms are available on an experimental basis.

2 The case for a Virtual Machine Logbook

Less than two months from data-taking, the ATLAS software system is now heavily used to analyze the behaviour of the detector in its commissioning stage, and to tune up data analysis strategies on simulation data. At the same time there is still a lot of development work ongoing, from tuning of trigger algorithms, to performance monitoring tools, to literally hundreds of bug fixes, and missing or late features. As a consequence it is not uncommon for an ATLAS physicist to work on several tasks at the same time. Context switches among different tasks are time-consuming even for the experienced developer. Sharing the work environment among ATLAS colleagues is even less trivial: it is not uncommon to spend several days exchanging emails before one can reproduce an interesting result or a problem reported by a colleague.

Today most code development and preliminary analysis is performed on the central CERN linux cluster (lxplus), but the resources available at CERN Tier 0 will not be sufficient to support every ATLAS physicist once data taking starts. The ATLAS computing model[4] calls for a multi-tiered, distributed approach to computing, with data analysis performed mainly on smaller regional centers and on local clusters and even personal desktops/laptops. While the larger regional centers will offer extensive support and will be functionally equivalent to CERN Tier-0 facility, physicists at many universities do not have the capabilities or time to install and mantain ATLAS software. They often have access to considerable hardware resources, but since those are shared among different experiments and projects, they are not running the standard ATLAS SLC4/gcc platform. Getting ATLAS software to run on these facilities is not trivial and often impossible.

Even when software issues are resolved, most of ATLAS data will not be accessible from regional centers and certainly not from a physicist laptop. An analysis developed at a local university needs to be packaged and carefully tested before it can be distributed on the grid for mass processing. Conversely a physicist trying to understand in detail the results of a central processing, needs to be able recreate the software environment of that processing, months of years after it occurred.

Proposed Project Definition

The goal of this project is to provide a working, if not full-featured, version of the Virtual Machine Logbook with sufficient quality to be distributed to early adopters that will provide feedback on usability and real-life performance.

The Virtual Machine Logbook will address the use cases described in the introduction allowing ATLAS physicists to easily switch between different development and analysis tasks, migrating their work from CERN lxplus, to their regional data centers, to their laptops. Entries in the VML should be shareable among users to allow reproducing an interesting result or a crash. Finally VML entries should be packaged in such a way that they can be shipped across the grid for user or group-level processing.

Besides the VML engine and UI, for VML to be functional we need to investigate/develop a technology independent mechanism to share files among the VMs, their hosts, and to access mass storage systems containing ATLAS experimental data, presumably using GRID data transfer tools.

Core Features

We expect these features to be available for beta-testing at the end of the project

1. Develop a VML engine and a user interface (both CLI, and browser based) that allows to manage logbook entries and to export them.

2. The UI should allow to start multiple instances of a VM in the logbook on a remote server/cluster/cloud. To parallelize execution of python scripts one can use python extensions like http://www.parallelpython.com/. As part of the CernVM project, we could build a cluster of virtual machines supporting such computation and running `ppserver.py` using autodiscovery mechanism and perhaps complement by developing some kind of a connection broker and scheduler to allow multi user access.

3. CernVM incorporates a http-based, read-only file system to provide file-by-file access to project-speficic software. We should investigate user-friendly and efficient mechanisms to package this project software to download it from CernVM repositories to the VMs.

4. Investigate various possibilities and choose optimal way of sharing file system between physical host and virtual machines running as a guest. Develop tools to simplify and automate such file sharing regardless of virtualization technology being used.

5. If the data is inside Host Machine, compare the performance of different ways to share these data to the guest. After choosing the best performed approach, design and implement an easy to use interface to Enable/Disable these shares and to access the data in VM.

Further Work

These are ideas that we may pursue during the project if time allows

1. investigate mechanism to generate VML entry from a non-VM environment. Notice that what is needed is a mechanism to capture the software environment (environment variables, software releases, local work areas), not a full snapshot of a physical machine to a VM.

2. investigate mechanism to efficiently add the status of relevant shared file system to a VML entry so that it can be later restored or shared with a colleague.

3. Investigate ways to access mass storage systems from virtual machine in various scenarios (with and without NAT). Carry our performance tests and come up with list of recommendations. If necessary, develop tools to allow virtual machines to access all mass storage systems even if they run behind NAT.

4. Investigate the possibility of accessing data on the GRID directly inside VM.

5. Investigate the feasibility of using a non-static (but still read-only) file-system, capable to synchronize automatically the local disk cache with the status of the CernVM software repository.

Preferred Tools and Boundary Conditions

ATLAS has started an investigation of Virtual Machine technologies, and it is currently collaborating with the CernVM project[5]. Therefore we should favour tools which are supported by the CernVM project. This is not really a strong constraint since CernVM is based on rBuilder a Virtual Appliance build system which supports all major VM platforms (QEMU/KVM, Parallels, XEN, and VMware).

Another constraint in our choice of technologies is ATLAS requirement to rely on open-source or at least freely available software products, especially for applications that will potentially be used by a large fraction of the collaborators.

Lastly the tools we adopt should provide a virtualization solution that works on Linux hosts, but also on Windows and Mac OS X.

Acknowledgements

We are grateful to CernVM project leader Predrag Buncic (CERN) for his suggestions.

Glossary

Based on [6, 7, 8].

Event What occurs when two particles collide or a single particle decays. Particle theories predict the probabilities of various possible events occurring when many similar collisions or decays are studied. They cannot predict the outcome for any single event.

Offline Software the software used to simulate, reconstruct, and analyze data after they have been recorded on disk and tape for the first time

Online Software the software used by the Trigger and Data Acquisition systems to process the data as they come out of the detector. In ATLAS, a large and increasing amount of core and algorithmic software is used both online and offline.

Trigger A set of algorithms implemented both in hardware and software that are used to filter out non-interesting events, thereby reducing the data-flow rate coming from the detector to manageable levels.

References

[1] http://atlas.web.cern.ch/Atlas/index.html

[2] https://twiki.cern.ch/twiki/bin/view/Atlas/CoreSoftware

[3] http://physics.bu.edu/~youssef/pacman/index.html

[4] http://atlas-proj-computing-tdr.web.cern.ch/atlas-proj-computing-tdr/PDF/Computing-TDR-final-July04.pdf

[5] http://cernvm.cern.ch/cernvm

[6] http://atlas.ch/glossary/glossary.html

[7] http://wlav.web.cern.ch/wlav/athena/athask/glossary.html

[8] http://documents.cern.ch/cgi-bin/setlink?base=atlnot&categ=PUB&id=gen-pub-2008-001